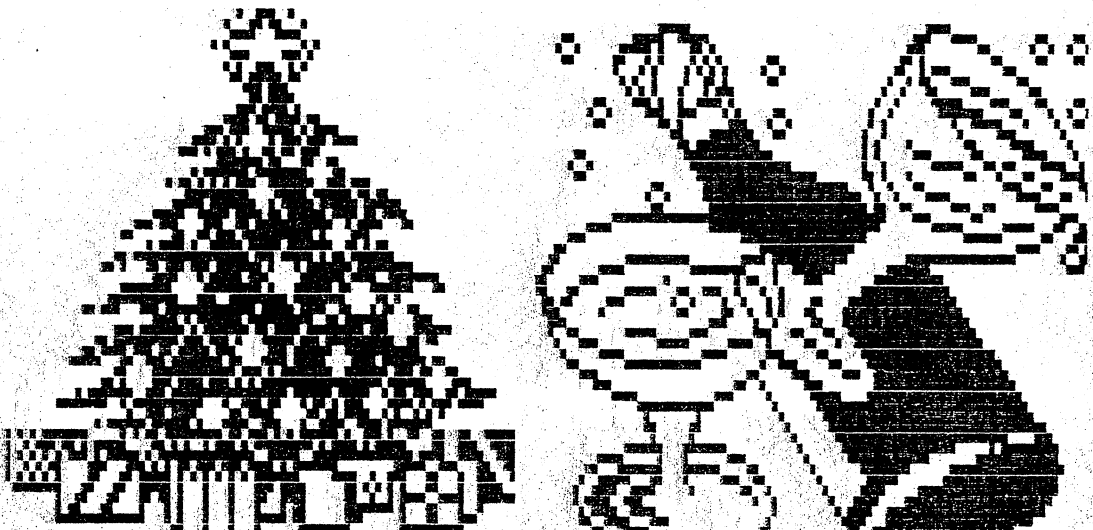
NOV/DEC 1987 ISSUE # 15

Hunter Valley VZ Journal



PRODUCED BI-MONTHLY BY H.V.VZ.U.G.

VZ USER GROUPS AND PUBLICATIONS :- Page 3

PRINTER PATCH V1.4 REVIEW :- Pages 4-5

This latest version of Larry Taylor's Printer Patch has a few enhancements over previous version. So if you have an EPSON or EPSON compatible printer then this Patch will print out all the VZ's inverse and graphic character set and do both LO and HI-RES screen dumps as well.

EDUCATIONAL - MENTAL MATHS by John GARLAND :- Pages 5-6

With quite a few VZ's appearing under the christmas tree this year the maths program would allow school children to do more than play games on their VZ.

KALEIDOSCOPE by Robert QUINN :- Page 7

If watching all the soap operas on TV leaves you bored then the endless patterns in KALEIDOSCOPE will dazzle you.

HI-RES CSAVE/CLOAD ROUTINES by Dave MITCHELL :- Pages 8-9

Disk drive users used to have it all to themselves being able to SAVE/LOAD HI-RES screens, but no more thanks to Dave. Not as fast as disk but it does it's job and you can have your own slide show.

MAJOR MAILING LIST UPDATE by Joe LEON :- Pages 10-13

As promised in last issue this update adds host of disk commands plus couple other features as well. Now you can have many DATA FILES on the one disk instead of just one as was the case previously. Further improvements can be made and I leave these to you. My thanks to Staff at D.Smith for their help with this article.

BOOK REVIEW by Ross woods :- Page 14

Ross gives a brief rundown on an Assembly Language Programming Manual for beginners written by Steve Olney. It is specifically written for the VZ so it should be very helpfull for beginners and advanced programmers alike. Don't forget the book can only take you so far and what you get out of it depends on the effort you put in.

BLOCK TRANSFERS by Chris HOBROUGH :- Pages 15-16

Last issue Robert Quinn showed us how to do block transfers from basic while in this issue Chris shows how to do it in Assembly Language. Steve Olney's book could be a big help in understanding the procedures involved.

ENHANCING FIND ROUTINE by Larry Taylor :- Pages 17-19

I found the String search routine to be very usefull as it will FIND both numeric and literal strings in your basic programs. Next issue will have a basic version for those persons not familiar with assembly language which is most of us.

DO IT YOURSELF :- Page 19

If some of the Keys on your VZ200/300 don't seem to work the you may need a new membrane keyboard which most persons should be able to change themselves.

FOR SALE - E & F W.P. PATCH3.1 and EXTENDED DOS V1.0 - Page 20

NEXT ISSUE - 8K BIB RAM PART III :-

BELIEVE IT OR NOT :-

This is a story about four people named EVERYBODY, SOMEBODY, ANYBODY and NOBODY. There was an important job to be done and EVERYBODY was sure SOMEBODY had done it. ANYBODY would have done it but NOBODY did it. SOMEBODY got angry about that because it was EVERYBODY's job. EVERYBODY thought ANYBODY could do it but NOBODY realised that EVERYBODY did not do it. It ended up that EVERYBODY blamed SOMEBODY because NOBODY did what ANYBODY could have done.

The Committee of Hunter Valley VZ Users' Group wishes all our members and their families both near and far the MERRIEST of CHRISTMAS and HAPPINESS and PROSPERITY for the NEW YEAR.

Another year has nearly gone and it's a nice feeling knowing we'll still be around next year. Once again I would like to thank our members both near and far for their continued support.

My thanks also go to Dave BOYCE, Dave MITCHELL, Robert QUINN and Larry TAYLOR without whose continued support we would'nt have much of a publication. Thanks Guys. I'm constantly amazed at the time and effort some members put in when asked for help or advice.

This issue also marks my first year as Editor. I found the work very time consuming, personally rewarding and at times frustrating, never more so than the last month or so. At times I doubted whether this issue would be published this year.

I had major system failures since my son (yes him again) decided to plug in his C64 in my little black box with surge surpressors, etc. My VZ has'nt been the same since then with constant crashes, hangups, wiping disks, etc. My son promised he would'nt touch my VZ again. He even made a New Years resolution - HAVE SCREWDRIVER - WILL TRAVEL.

VZ USER GROUPS - PUBLICATIONS . .

VZ USER MARK HARWOOD P.O.BOX 154 DURAL N.S.W. 2158

LE'VZ OOP J.C.E. D'ALTON 39 AGNES St. TOOWONG QLND. 4066
VSOFTWAREZ - SOFTWARE/HARDWARE FOR SALE

VZ DOWN UNDER SCOTT LE BRUN 59 BRENTWOOD DVE WANTIRNA 3152
SOFTWARE/HARDWARE FOR SALE

VZ-LINK - PETER J. HILL P.O.BOX 1972 C.P.O. AUCKLAND N.Z.

WAVZ - GRAEME BYWATER P.O. BOX 388, MORLEY W.A. 6062

BRISBANE VZ USERS WORKSHOP - C/O 63 TINGALPA ST. WYNUM WEST 4178

HUNTER VALLEY VZ USERS' GROUP - P.O. BOX 161 JESMOND N.S.W. 2299
EDITOR-JOE LEON (049)51 2756 - SECRETARY-ROSS WOODS (049)71 2843

SUBSCRIPTION - H.V.VZ.JOURNAL - 6 MONTHS $9.00 - 12 MONTHS $18.00
New Zealand - 6 MONTHS $12.00 - 12 MONTHS $24.00

NEW VENUE - NEW DATES - NEW VENUE - NEW DATES - NEW VENUE

MEETINGS - FIRST FRIDAY of MONTH at JESMOND COMMUNITY CENTRE
MORDUE PARADE - REAR STOCKLAND MALL (BIG W) JESMOND

NOTE :- When writing to any above or H.V.VZ. Users' Group for information please enclose a S.S.A.E. or 2 Int. Reply Coupons.

No MATERIAL in this Journal may be reproduced in part or whole without the consent of the Author who retains COPYRIGHT.

VZ-EPSON PRINTER PATCH V1.4 --- AVAILBLE FROM VSOFTWAREZ

The VZ-EPSON PRINTER PATCH allows EPSON or EPSON compatible printers to LPRINT or LLIST all of the VZ's INVERSE and GRAPHIC characters. The LO-RES and HI-RES screens can also be dumped to the printer as well.

This latest Version 1.4 has a few extra enhancements over previous versions which I found very useful. This Journal would look dull if I didn't make use of the facilities offered by the Patch. All the program LISTings, screen dumps of both HI and LO-RES screens were done by this Patch.

Before the Patch like many VZ users I used to go right through most programs and remove all INVERSE and GRAPHIC characters so my printer would print out the programs. The only printer on the market that used to be able to print out the VZ character set was the GP 100 which is no longer availble.

INVERSE and GRAPHIC CHARACTERS :-

The Patch handles them with ease with improved character form. Have a look at Mental Maths and Mailing List Update programs for examples.

LO AND HI-RES SCREEN DUMPS :-

As there are many different printers on the market which sometimes don't print out the same and have different LINE FEED characteristics. For that reason the COPY command was given extra parameters, COPYn and COPYAn, where n can be from 4 to 8.

The COPYA command is very usefull for LO-RES screen dumps and is ideal for printing program menus. When typing in a menu from a program sometimes it's a bit hard to know what it should look like. The Mental Maths and Mailing List Update screen dumps of their respective menus show what the menus should look like. With my AMSTRAD DMP-2000 printer I used COPYA8 for LO-RES menu screens.

NOTE - A one dot wide border is printed to denote borders.

COPYA5 was used for HI-RES screens. See Kaleidoscope program for examples of HI-RES screen dumps which were reduced to fit the page. The screens dumps are correct as Kaleidoscope produces vertical patterns which only use middle part of screen.

LTAB(n) - PAGE(n) - FEED(n) :-

These 3 commands are very useful with program LISTings.

LTAB(n) - n = 0 to 63 - This command is used for setting of left margin and works with LLIST, LPRINT and COPY commands. It can also work with DIR and STATUS as well by directing the VZ to print to the printer instead of the screen. EG. :-

```
LTAB(10):POKE 30876,1:DIR:STATUS
```

If you want to use another TAB position then just type in your new TAB setting. There is no need to zero your previous TAB setting as the LTAB(n) command does it for you automatically. LTAB(45) is the same as LTAB(0):LTAB(45).

PAGE(n) - n = 0 to 127 - FEED(n) - n = 0 to 127 :-

These two commands set the PAGE length (Number of lines per page) and perf skip. FEED(n) will be ignored unless it is less than page length. The 3 commands used together are ideal to LLIST your programs. EG. :-

LTAB(10):PAGE(66):FEED(6):LLIST

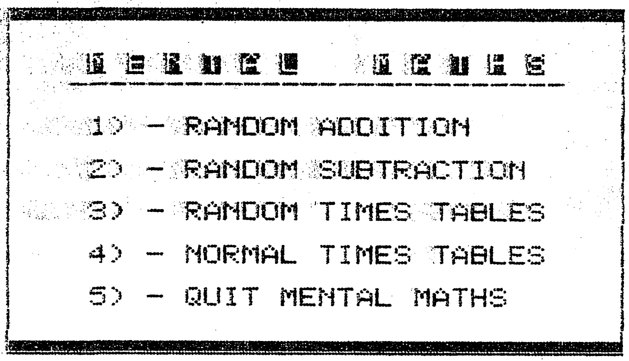
Using my printer and computer paper the above LLISTS my program to the printer with a left margin of 10, 66 lines per page and perf. skip of 6 lines. This assures me that no text is printed on or near the perforations. The setting for your printer and paper may be different than mine. Test out all the functions of the patch so you understand and are familiar with it's operation.

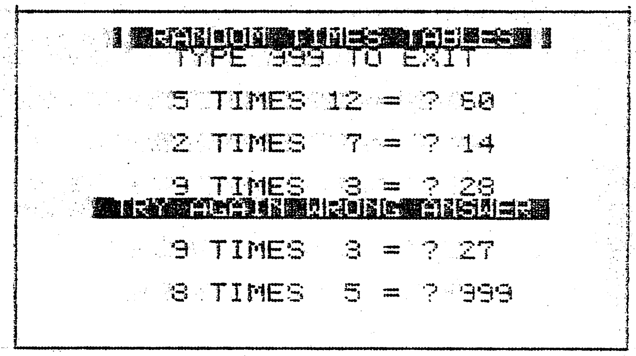
It's good practice to ZERO the PAGE and FEED commands after use as it's easy to forget or you could end up with a form feed in the middle of the page. Setting FEED(0) to zero turns of auto form feed.

LTAB(n), PAGE(n), FEED(n), LLIST, LPRINT and both COPY commands can be used in direct mode or from within your program, but will only work properly while the patch is present. The Patch will also recognise all the hidden commands which may have been entered using an extended basic utility. Steve Olney's EXTENDED BASIC can also be used with the patch, but is not dependant on it.

VERDICT :- Top marks to Larry Taylor for an exellent utility. I could'nt imagine being without it. Ed.

EDUCATIONAL - MENTAL MATHS . . .





The Mental Maths program on the next page was designed for young children to learn simple maths. For that reason the addition and subtraction top limit of 20 was chosen.

The largest sum possible is 20 plus 20 or 20 minus 20. It's not too difficult to to raise or lower upper limit on all four routines. I'll explain how it can be accomplished using the Random Addition routine at lines 100 to 160 as an example.

Line 110 PRINT:A=RND(20):B=RND(20)
Change the 20 in both enclosed brackets to 99.
Line 120 IFA> 20 ORB> 20 THEN110
Change both 20's in line 120 to 10.

What we have done is raise the upper limit in line 110 to 99 while lowering lower limit in line 120 to 10. Line 120 tests to see if the randomly selected number is greater than 10 and if it is it sends it back to line 110 to select another number. This test is carried out till the number selected is less than or equal 10. Instead of upper limit of 10 a larger upper limit can be selected.

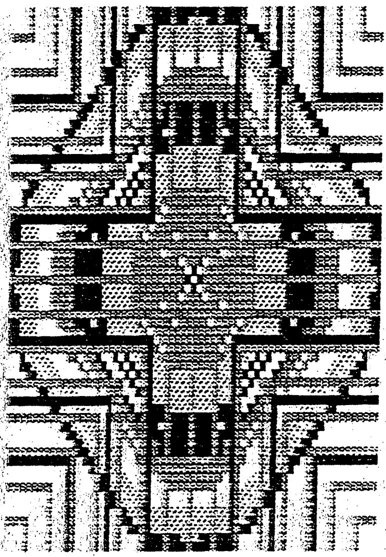
ENHANCEMENTS :-

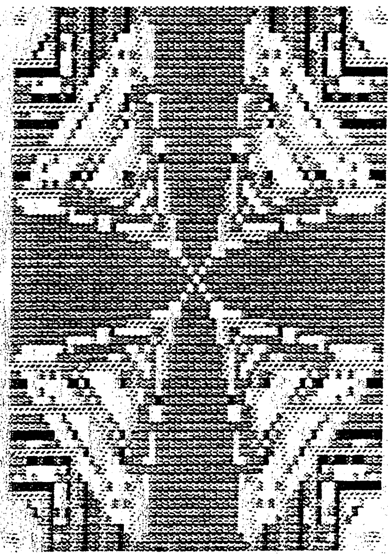
Although the program can be tailored for a particular level it would be more desirable to include in the program level selection. That way this program could be used over a wider age group. Other enhancements could be made as well to suit your particular needs. Our aim was to give you a basis to work from.

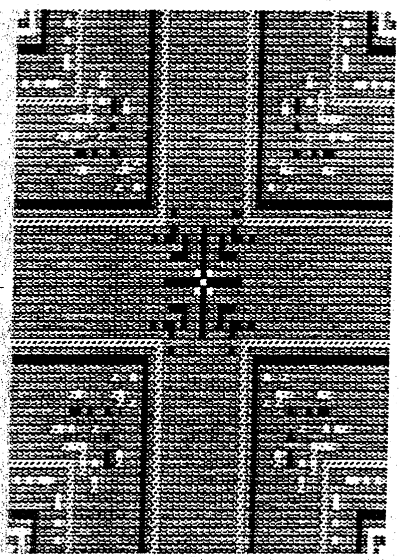
```
10 '***************************************************************
11 '*  MENTAL  MATHS  ROUTINES  SUPPLIED  BY  JOHN  GARLAND  *
12 '***************************************************************
13 :
14 CLS:COLOR,0:POKE30744,96:COLOR7:POKE30862,80:POKE30863,52
15 PRINT"████████████████████████████████";:PRINT:'32 SHIFT 'T'
20 PRINT"    ▇▇▇▇▇▇  ▇▇▇▇▇"
25 PRINT"    ----------------------------":PRINT
30 PRINT"    1) - RANDOM ADDITION     ":PRINT
35 PRINT"    2) - RANDOM SUBTRACTION ":PRINT
40 PRINT"    3) - RANDOM TIMES TABLES":PRINT
45 PRINT"    4) - NORMAL TIMES TABLES":PRINT
50 PRINT"    5) - QUIT MENTAL MATHS   ":PRINT
55 PRINT"███████████████████████████████";:POKE29183,227
60 GOSUB600:REM 31 SHIFT 'Y'
65 A$="":A$=INKEY$:A$=INKEY$:IFA$=""THEN65ELSEX=USR(X)
70 IFA$="1"THEN100ELSEIFA$="2"THEN200
75 IFA$="3"THEN300ELSEIFA$="4"THEN400
80 IFA$="5"THENCLS:ENDELSE65
90 :
100 CLS:PRINTTAB(6)"▌RANDOM ADDITION ▌":GOSUB700:GOSUB600
110 PRINT:A=RND(20):B=RND(20)
120 IFA> 20 ORB> 20 THEN110
130 PRINTTAB(6)USING" ##";A;:PRINT" PLUS";
140 PRINTUSING" ## ";B;:PRINT"=  ";:INPUTK
150 IFK=999THENRUNELSEIFK=A+BTHEN110
160 IFK<>A-BTHENGOSUB500:PRINT:GOTO130ELSE110
190 :
200 CLS:PRINTTAB(5)"▌RANDOM SUBTRACTION ▌":GOSUB700:GOSUB600
210 PRINT:A=RND(20):B=RND(20)
220 IFA> 20 ORB> 20 OR A<B THEN210
230 PRINTTAB(6)USING" ##";A;:PRINT" MINUS";
240 PRINTUSING" ## ";B;:PRINT"= ";:INPUTK
250 IFK=999THENRUNELSEIFK=A-BTHEN210
260 IFK<>A-BTHENGOSUB500:PRINT:GOTO230ELSE210
290 :
300 CLS:PRINTTAB(5)"▌RANDOM TIMES TABLES ▌":GOSUB700:GOSUB600
310 PRINT:SOUND30,1:A=RND(12):B=RND(12)
320 IFA> 12 OR B> 12 THEN310
330 PRINTTAB(6)USING" ##";A;:PRINT" TIMES";
340 PRINTUSING" ## ";B;:PRINT"= ";:INPUTK
350 IFK=999THENRUNELSEIFK=A*BTHEN310
360 IFK<>A*BTHENGOSUB500:PRINT:GOTO330ELSE310
390 :
400 CLS:PRINTTAB(4)"▌NORMAL TIMES TABLES ▌":GOSUB700:GOSUB600
410 PRINT:FORA=1TO12:FORB=1TO12:SOUND30,1
420 IFA> 12 OR B> 12 THEN410
430 PRINTTAB(6)USING" ##";A;:PRINT" TIMES";
440 PRINTUSING" ## ";B;:PRINT"= ";:INPUTK
450 IFK=999THENRUNELSEIFK=A*BTHEN470
460 IFK<>A*BTHENGOSUB500:PRINT:GOTO430ELSE410
470 NEXTB:NEXTA
490 :
500 COLOR,1:PRINTTAB(4)"TRY AGAIN WRONG ANSWER":SOUND30,1;1,2
510 COLOR,0:RETURN
600 SOUND30,1;25,1:RETURN
700 PRINTTAB(8)"TYPE 999 TO EXIT":RETURN
```

```
200 '*************************************************************
210 '*   KALEIDOSCOPE PROGRAM DESIGNED BY ROBERT QUINN           *
220 '*************************************************************
230 '*   N - SWITCH BACKGROUND COLOR    A - CLEAR THE SCREEN     *
240 '*   B - SELECT BLUE                G - SELECT GREEN         *
250 '*   Y - SELECT YELLOW              R - SELECT RED           *
260 '*   SPACE - PAUSE SWITCH           RETURN - RESTART         *
270 '*   0 - 9 - SELECT PARAMETER       Q - ADD PARAMETER TO X   *
280 '*   W - ADD PARAMETER TO Y         C - COPY TO PRINTER      *
290 '*************************************************************
300 MODE(1):H=3:COLOR,0
310 W=1:F=1:C=1:T=4:G=2:GOTO340
320 H=H+1:IFH=5THENH=1
330 C=NOTC:W=RND(9):F=RND(9):T=RND(9):G=RND(9):IFRND(6)=6THEN310
340 COLORH:R%=1:A=63:B=31
350 X=0:Y=R%:E=1-R%:U=1:V=1-2*R%
360 IFNOT(X>Y)THEN370ELSER%=R%+1:IFR%<32THEN350ELSE320
370 SET(X+A,Y+B):SET(Y+A,X+B):SET(Y+A,-X+B):SET(X+A,-Y+B)
380 SET(-X+A,-Y+B):SET(-Y+A,-X+B):SET(-Y+A,X+B):SET(-X+A,Y+B)
390 A$=INKEY$:A$=INKEY$:IFA$=""THEN460ELSESOUND20,1
400 IFA$>"/"ANDA$<":",L=VAL(A$):ELSEIFA$="N",K=NOTK:COLOR,ABS(K)
410 IFA$="A"THENMODE(1)
420 IFA$="W"THENW=LELSEIFA$="Q"THENF=L
430 IFA$="G"THENH=1ELSEIFA$="Y"THENH=2
440 IFA$="B"THENH=3ELSEIFA$="R"THENH=4
445 IFA$="C"THEN:COPYA5
450 COLORH:IFA$=CHR$(13)THEN310ELSEIFA$=" "THENJ=NOTJ
460 IFJ<0THEN390
470 IFE<0THENV=V+RND(9):IFC=1THENE=E+U:GOTO490ELSEE=E+V:GOTO490
480 Y=Y-W:V=V+T:E=E+V
490 X=X+F:U=U+G:GOTO360
500 :
```

EDITOR'S COMMENT :- When typing in above program there's no need to type in lines 200 to 290 or line 500. This program would be ideal for tape users to try it with Dave Mitchell's HI-RES CLOAD/CSAVE routine. It will fit nicely between lines 170 to 1000 in listing 3. Line 445 is for screen dumps to printer for persons having Larry Taylor's Printer Patch V1.3 or V1.4. Below are 3 screen dumps which have been reduced.







There are a few routines around that save and load screen dislpays from tape but they involve moving the displays around in memory using basic. They do but are a bit messy where you CLOAD the display but after it is loaded the CLOAD routine jumps to basic and does not return to the program you are running. You then have to run the next line number after CLOAD.

What should be done is to run the program without any breaks. This is what I have achieved. The program CLOADS MODE (1) displays from tape and loads directly to the screen in the same way as BLOADING does. BLOADING screens takes aprox. 4 seconds and CLOADING takes aprox. 30 seconds after the display has been found on tape. This may seem long as compared with disk but it is quicker than the other versions I discribed earlier.

I use a small machine code routine to capture the command CLOAD and it is then processed by my new CLOAD routine. This new routine then loads from tape ONLY BASIC PROGRAMS that start at 28672 decimal (7000H) ALL OTHER PROGRAMS ARE IGNORED. So after using this program the computer should be reset to use CLOAD in the normal manner.

LISTING 1 :-

```
10 POKE30884,151:POKE30885,123
20 POKE30969,153:POKE30970,123
```

Type in listing 1 and run. Listing 1 sets the start and end of program to a new start and end to allow for the machine code to be placed at the start of the program so as to allow the machine code to be saved along with the basic program. Normally the start of a basic program is 31465 decimal, it is now shifted to 31639.

Line 10 shifts the start address
Line 20 shifts the end address

LISTING 2 :-

```
10 FOR I = 31463 TO 31639:READ A :POKE I,A:B=B+A:NEXT
20 IF B <> 19235 THEN PRINT "ERROR IN DATA":END
30 NEW
40 DATA 0,0,9,123,0,0,177,51,48,56,54,50,44,49,51,58,177,51,48
50 DATA 56,54,51,44,49,50,51,58,88,213,193,40,88,41,0,0,0,0,,0
60 DATA 42,4,120,34,69,123,33,42,123,34,4,120,17,151,123,237
70 DATA 83,164,120,33,38,123,195,233,54,142,0,0,0,217,33,91,29
80 DATA 209,183,237,82,213,217,194,120,29,229,205,120,29,32,2
90 DATA 209,201,254,185,40,4,225,195,147,66,35,209,209,17,30,29
100 DATA 213,229,33,57,120,203,182,203,158,225,243,205,140,53,229
110 DATA 205,177,53,33,66,56,205,244,55,205,231,53,58,210,122
120 DATA 254,240,32,246,33,96,56,205,4,56,221,33,35,120,205
130 DATA 104,56,56,231,122,254,112,32,226,205,115,63,245,122
140 DATA 254,120,40,5,241,18,19,24,242,241,225,251,201,0,0,0
```

Type in listing 2 and RUN. LISTING 2 is for the capture and execution of the command CLOAD.

LISTING 3 :-

```
100 MODE(1)
110 POKE31273,PEEK(30884):POKE31274,PEEK(30885)
120 POKE31275,PEEK(30969):POKE31276,PEEK(30970)
130 A$="":A$=INKEY$:A$=INKEY$:IFA$=""THEN120
140 IFA$="S"THENSOUND20,1:GOSUB1000
```

```
150 IFA$="L"THENSOUND20,1:GOSUB1050
160 IFA$="C"THENMODE(1)
170 GOTO120
1000 POKE30884,0:POKE30885,112:POKE30969,0:POKE30970,120
1010 CSAVE"PICTURES"
1020 POKE30884,PEEK(31273):POKE30885,PEEK(31274)
1030 POKE30969,PEEK(31475):POKE30970,PEEK(31476)
1040 SOUND20,1:RETURN
1050 POKE30796,1:CLOAD
1060 POKE30796,0:SOUND20,1:RETURN
```

Type in listing 3 but do not run. Listing 3 is a small routine for CSAVING and CLOADING from tape. It is only the bare essentials. Screen dumps to printers, etc I leave to someone esle. Disk users could add disk saving and loading features to it as well. The following is how I transfered screen dumps from disk to tape.

```
1 GOTO 100
10 MODE(1):BLOAD"PICTURE"
```

Everytime I wanted to save a display from disk I typed the name of the display into line 10 and RUN 10. NO instructions appear on the screen as it goes to HI RES, (MODE 1).

S key SAVES the display to tape
L key LOADS a display from tape
C key clears the MODE(1) screen

LISTING 3 EXPLANATION :-

LINE 100 MODE(1) HI RES.
LINE 110 saves the start address of the program
LINE 120 saves the end address of the program
LINE 130 - 170 looks for an input from the keyboard
LINE 1000 sets up the start & end addresses for saving the screen
LINE 1010 CSAVES
LINE 1020 -1030 resets the start & end addresses
LINE 1040 makes a noise and returns for another input from the keyboard
LINE 1050 POKE 30796,1 disables the waiting & loading messages at the bottom of the screen. the only tape message printed if it happens is the loading error.
LINE 1060 resets 30796 & makes a noise & returns for another keyboard input.

The sound commands in the routine are important as we have no other way of knowing if the loading and especially the saving routines are finnished.

After entering listing 3 we are almost ready to csave the program but first we must change the start address back to 31465 WITH NO LINE NUMBERS TYPE :-

POKE 30884,233:POKE 30885,122 and then CSAVE or SAVE the program.

The CLOAD (machine code) routine is saved along with the basic program Entering more lines is no problem just type away. OR after you have a number of screens saved on tape you could use a FOR NEXT loop to load all the displays one after another.

EG. :- POKE30796,1:MODE(1):FORI=1TO N :CLOAD:NEXT
N is the number of displays saved on tape.

This and the following two pages contain the changes and additions to be done to Mailing List to give disk users full compliment of disk commands. The most important being SAVE which allows you to save DATA FILES to disk using any filename up to 8 characters long. What it means is that now you can save many DATA FILES on disk instead of only one as per D.Smith's Technical Bulletin 111.

Whether you have original Mailing List or modified version according to Bullettin 111 the following mods are same for both versions and now to the conversions. If you have an Extended Basic with delete command then BRUN it first and then load Mailing List into the VZ.

STEP 1 - LIST line 20. It should read - 20 GOTO33 or 20 GOTO35. Next cursor up to line 20 and change 20 to 10. This new line 10 is the first modification and do not change it again as shown on next page. I bought my M/List years ago and when I compared it to the version being sold now line 20 was one of the differences I discovered. Therefore your new line 10 must GOTO where your previous line 20 went otherwise the mods wont work.

Before typing in the new lines some old and no longer neccessary lines must be deleted. Type in the following six lines one at a time pressing RETURN after each line number. 12 14 20 1085 1205 1300

Next a range of lines must be deleted. If you have Ext. Basic loaded then use the DELETE command with the following lines otherwise use the little delete routine below. 1000-1040 5000-5240 6000-7090 23000-23040

```
0 D1000-1040
2 POKE31469,68:LIST 0
4 POKE31469,182:GOTO 0
```

Next type in RUN4 and press RETURN and then type in RUN2 again pressing RETURN. If you try to LIST lines 1000-1040 you'll find they no longer exist. LIST line 0 and type in the next range of lines to be deleted and repeat previous procedure. Continue till all required lines have been deleted. Next delete above routine by typing 0 and pressing RETURN and the same for line 2 and 4.

Now we can proceed to enter the new lines and care must be taken with them because if you type in wrong line number you could wipe out existing line.

DO NOT CHANGE, ADD or SUBTRACT a single character or space in lines 10 to 20 as these are the heart of the new disk routines. All the spaces within the quotes in above lines are 8 spaces long. When saving program to disk save it as DISKLIST.

The operations of the ERAse, REName, SAVE and LOAD routines are the same so any comments made about the ERAse routine also apply to the others as well.

ERASE ROUTINE (Lines 800-805) :-

Line 800 - INPUT N$ asks for FILENAME.
GOSUB770 - This is SPACE TO START 'Q' TO QUIT routine.
AD=31481 - This is the memory address of the first SPACE in line 12.
GOSUB22 - This is the FILENAME POKE routine. It POKEs the FILENAME you entered between the quotes in line 12. AD is the the address it pokes it in. If your filename is less than 8 characters then the POKE routine adds extra spaces to make up 8 characters.
GOSUB 12 - Line 12 does the actual ERAsing.

NOTE - If you list lines 10 to 20 after using ERA, REN, SAVE and LOAD functions you'll see previous filenames printed between the quotes. Take no notice of them as the NEW FILENAME writes over them.

```
10 GOTO35
12 ERA"          ":RETURN
14 REN"          ","          ":RETURN
16 OPEN"          ",1:PR#"          ",DT,DT$:RETURN
17 PR#"          ",D$(N):RETURN
18 OPEN"          ",0:IN#"          ",DT,DT$:RETURN
19 IN#"          ",D$(N):RETURN
20 CLOSE"          ":RETURN
21 '"DON'T CHANGE LINES 10-20
22 FORPO=0TO7:POKEAD+PO,ASC(MID$(N$+"        ",PO+1,1)):NEXT
23 RETURN
24 :

46 MI$="============================":REM 28 EQUAL SIGNS
47 SP$="                            ":REM 28 SPACES
48 SQ$=" SPACE TO START / Q TO QUIT ":MK$="█"

600 GOSUB30:COLOR7:PRINT@34," DISK MENU     ";DT$
605 PRINT@66,MI$
610 PRINT@ 98,"R - REN  FILE  E - ERA  FILE"
615 PRINT@162,"S - SAVE DATA  L - LOAD DATA"
620 IFNE$="CHANGE"THENPRINT@163,MK$
630 PRINT@226,"D - DIRECTORY  I - INIT DISK"
635 PRINT@290,"M-MENU P-PEEK  &-SAVE ^-REPL":PRINT@322,MI$
640 PRINT@354,SP$:PRINT@386,SP$:PRINT@450,SP$
645 COLOR7:PRINT@354," SELECT FUNCTION         ":SOUND30,1
650 SF$="":DM$="":DM$=INKEY$:DM$=INKEY$:IFDM$=""THEN650
655 IFDM$="R"THEN820ELSEIFDM$="E"THEN800ELSEIFDM$="S"THEN880
660 IFDM$="L"THEN850ELSEIFDM$="D"THEN680ELSEIFDM$="I"THEN750
665 IFDM$="M"THEN1000ELSEIFDM$="^"THEN920
670 IFDM$="&"THEN935ELSEIFDM$="P"THEN700ELSE650
675 :
680 COLOR7:SF$=" DISPLAY DIRECTORY ":GOSUB770
685 CLS:DIR:STATUS:PRINT@17," PRESS SPACE ";
690 SOUND30,1:GOSUB780:GOSUB7000:GOTO600
695 :
700 REM"MEMORY PEEK ROUTINE
705 CLS:DM$="":PRINT@2,SQ$
710 FORPK=31477TO31682:PRINTPK;PEEK(PK);CHR$(PEEK(PK))
715 IFINKEY$=" "THEN720ELSEIFINKEY$="Q"THENGOSUB40000:GOTO600
720 IFINKEY$=""THEN715ELSENEXT
725 SOUND30,1:COLOR7:GOSUB28:GOTO600
730 :
750 SF$=" INITIALISE DISK ":GOSUB770
755 PRINT@450," PLEASE WAIT 75 SECONDS ";:SOUND30,1
760 INIT:GOTO685
765 :
770 COLOR7:PRINT@354,SF$:SF$=""
775 PRINT@450,SQ$;:SOUND30,1
780 IN$=INKEY$:IN$=INKEY$:IFIN$=""THEN780
785 IFIN$=" "THENRETURNELSEIFIN$="Q"THEN640ELSE780
795 :
800 PRINT@354," ERA FILENAME   ";:SOUND30,1:INPUTN$:GOSUB770
805 AD=31481:GOSUB22:PRINT@450,SP$:GOSUB12:GOTO685
815 :
820 PRINT@354," OLD FILENAME   ";:SOUND30,1:INPUTN$
825 AD=31501:GOSUB22
830 PRINT@386," NEW FILENAME   ";:SOUND30,1:INPUTN$
835 AD=31512:GOSUB22:GOSUB770
840 PRINT@450,SP$:GOSUB14:GOTO685
845 :
```

```
850 PRINT@354,SP$
855 COLOR7:PRINT@354," LOAD DATA FILE   ";:SOUND30,1:INPUTN$
860 GOSUB770:PRINT@450,SP$;:SOUND30,1:AD=31603:GOSUB22
865 AD=31619:GOSUB22:AD=31646:GOSUB22:AD=31674:GOSUB22
870 GOTO6000
875 :
880 IFDT<>0THEN890ELSEPRINT@353,"     NO DATA IN MEMORY! "
885 SOUND5,7:GOTO645
890 PRINT@354," SAVE DATA FILE   ";:SOUND30,1:INPUTN$:GOSUB770
895 PRINT@450,SP$;:SOUND30,1:AD=31533:GOSUB22
900 AD=31549:GOSUB22:AD=31576:GOSUB22:AD=31674:GOSUB22
905 GOTO5000
915 :
920 SF$=" REPLACE PROGRAM ":GOSUB770:PRINT@450,SP$
925 COLOR7:PRINT@354," ERASING  : DISKLIST   ";:SOUND30,1
930 ERA"DISKLIST"
935 SF$=" SAVE PROGRAM    "
940 IFDM$="^"THEN945ELSEGOSUB770:PRINT@450,SP$
945 COLOR7:PRINT@354," SAVING  : DISKLIST    ";:SOUND30,1
950 SAVE"DISKLIST"
955 GOSUB7000:GOTO600
960 :
1000 IFDT$=""THENDT$="NO DATE"
1010 GOSUB30:COLOR7:PRINT@34," MAIN MENU        ";DT$:DM$=""
1020 PRINT@66,"NO. OF RECORDS IN FILE : ";:PRINTUSING"###";DT
1030 PRINT@98,MI$:PRINT@130,"1. ENTER DATA VIA KEYBOARD";
1040 PRINT@162,"2.  DOS DISK FUNCTIONS  ";
1045 IFNE$="CHANGE"THENPRINT MK$

1075 IFST$="SORT"THENPRINT MK$
1080 PRINT@290,"6. ENTER TODAYS DATE";

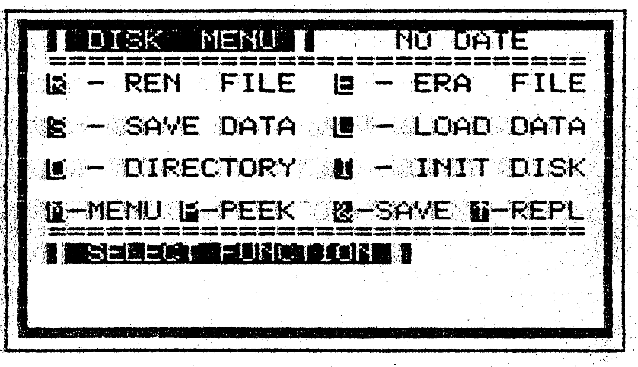
1110 PRINT@386,"9. PRINT MAILING LABELS":PRINT@418,MI$
1200 COLOR7:PRINT@450," WHICH FUNCTION (1-9) ?       ";:SOUND30,1
1210 K$=INKEY$:I$=INKEY$:IFI$=""THEN1210
1220 IFI$="6"THEN20000
1260 IFI$="2"THEN600

5000 REM"WRITE DATA TO DISK
5020 GOSUB16
5200 PRINT@354,"FILE CONSISTS OF";USING" ###";DT;
5210 PRINT" RECORDS"
5220 FORN=1TODT
5225 PRINT@418,"WRITING RECORD #";USING" ###";N
5230 GOSUB17
5240 NEXT:GOSUB20

5330 IFI$="1"THEN600ELSEIFI$="2"THENDT=0:ST$="":DT$="":GOTO600

6000 REM"READ DATA FROM DISK
6020 GOSUB18
6090 PRINT@354,"FILE CONSISTS OF";USING" ###";DT;
6100 PRINT" RECORDS"
6110 FORN=1TODT
6120 GOSUB19
6125 PRINT@418,"LOADING RECORD #";USING"###";N
6130 NEXT
6140 GOSUB20:GOSUB7000:GOTO600
7000 GOSUB40000:GOSUB30:COLOR7
7010 PRINT@166,"                          "
7020 PRINT@198," OPEN  DRIVE  DOOR "
7030 PRINT@230,"                          ":SOUND30,1
7040 IFDM$=""THENGOSUB27:RETURNELSEGOSUB28:RETURN
7090 :
```





MAIN MENU - As you'll notice by the MAIN MENU screen dump some changes have been made to it. The DATE status is printed at top of both Menus.

OPTION 2 - This now selects all the disk functions now.

OPTION 6 - The DATE can be entered anytime from the Main Menu.

OTHER OPTIONS - Refer to Mailing List instructions.

DISK MENU :-

R - RENAME FILE - This option is used to RENAME any disk files.

E - ERASE FILE - This option is used to ERASE any disk files.

S - SAVE DATA - This option is used to SAVE DATA files to disk using file name of your choice.

L - LOAD DATA - This option is used to LOAD DATA files from disk.

NOTE - FILENAMES can be from 1 to 8 characters long.

D - DIRECTORY - This option displays disk DIRectory and it's STATUS. Use this option before using any above.

I - INIT DISK - This option is used to INITialise disks.

M - MENU - This option displays MAIN MENU again.

P - PEEK - This option is used to find memory addresses for lines 10 to 20. If you want to change data in lines 10 to 20 then this routine will provide you with addresses required.

& - SAVE - This option will SAVE DISKLIST to disk.

^ - REPL - This option will ERASE DISKLIST from disk and REPLace (SAVE) with DISKLIST in memory.

NOTE - When most options are selected a QUIT option is provided in case you have to return to the MENU. The DATE is now also saved along with DATA FILE.

FILE NOT FOUND or FILE ALLREADY OPEN - If these ERRORS happen then DO NOT RUN DISKLIST as you will lose all the DATA. Instead put disk in drive, close door and type in the following to regain control without losing DATA.

GOTO 20 and press RETURN
GOTO685 and press RETURN

Line 20 will CLOSE OPEN FILE while line 685 will take you to the directory routine from where you can return to the menu.

VZ 200/300 ASSEMBLY LANGUAGE PROGRAMMING MANUAL FOR BEGINNERS.

This 140 page Manual has taken STEVE OLNEY over 18 months to write and is for sale at $24.95. I purchased my copy from :-

S.R.OLNEY P.O. Box 125 North Richmond N.S.W. 2754.

Below is a condensed version of the Contents to give some idea of the areas covered.

I found the Manual to be of a decent layout with a good index giving full details of each chapter, the Manual is well presented using a plastic ring binder.

Information is for the VZ not just Z80 type Computers and I'm sure anyone that USES this Manual in conjunction with the Dick Smith Electronics VZ 200/300 Technical Reference Manual will master Assembly Language Programming, Its quite obvious that one has to try out the routines and examples given in the Manual to grasp the subjects covered.

On my experience so far working through the Manual I was impressed with Steve's methods of explanation. Contents of each chapter are readily found in the contents list. I would recommend to anyone finding this Manual useful to purchase a copy of Programming The Z80 by Rodney Zaks Published By Sybex Computer Books, this book will allow you to cover the use of the complete instruction set to use and design useful routines.

(R.Woods.)

PS :- The Manual is also available from Dick Smith Electronics and ads should start to appear in various publications as well. Ed.

There are many situations in machine language programming where you need to move the contents of a whole block of memory to another location without processing it in any way. Data files, screen images and simple prompt messages all usually consist of a string of bytes which are stored in one place and then moved somewhere else to be used or displayed.

Ultimately, each byte must be moved individually but the Z-80 processor has four instructions which help speed things up. These are :-

```
LDI  - LoaD and Increment
LDIR - LoaD and Increment then Repeat
LDD  - LoaD and Decrement
LDDR - LoaD and Decrement then Repeat
```

Their use is best illustrated by the following example :-

```
LD HL, FROM     ;source
LD DE, TO       ;destination
LD BC, COUNT    ;counter
LDIR            ;automated transfer
```

This represents block transfer at it's simplest. First the pointers are set up, with HL containing the address of the first byte to be moved and DE containing the address of it's destination. Then BC is loaded with the number of bytes in the block. The last instruction then carries out the whole transfer. It moves the first byte then increments both HL and DE (HL=HL+1, DE=DE+1) to point to the next byte and decrements the count in BC (BC=BC-1). This is repeated until BC equals zero, thus moving the whole block. As you can see, it's a very powerful instruction.

An example of this in use is the scrolling of the screen which takes place as you list a program. Each time the bottom line is completed the whole screen must be moved up one line to make room for more at the bottom. To rewrite the entire screen, in its new position from the program table in memory would be a waste of time because the program is not stored in the form that appears on the screen and must be processed. It's much quicker to take the block of Video RAM which starts at the first byte of the second line and move it back 32 bytes to the start of the first line. The bottom line is then cleared and the next line of code processed and written in. As you can see, the whole screen is shifted in much less time than it takes to write the new bottom line.

Suppose you have a program which does something similar but you want to scroll the other way. If you used the routine above then FROM would be the start of the first line and TO would be the start of the second line. The result would be that the second line would be overwritten before it was moved and you'd end up with the first line repeated 15 times. The answer is to replace LDIR with LDDR and start at the bottom of the screen, at the end of the next to last line. This instruction, after moving each byte, decrements HL and DE as well as BC and so moves backwards through the block. If the size of the block is less than the distance being moved then it doesn't matter which instruction is used but if not then use LDIR to move down in memory and LDDR to move up.

Sometimes the exact length of the block is not known in advance. Records in a data file may be terminated with a carriage return as a marker and your block transfer routine will have to test for this marker to see if the end has been reached. In this case something like the following will be used :-

```
        LD   HL, FROM      ;set pointers
        LD   DE, TO        ;
        LD   BC, COUNT     ; & counter
        LD   A, 0DH        ; CR code for comparison
NEXT    CP   (HL)          ; compare character with CR
        JR   Z, END        ; exit if found
        LDI                ; else transfer byte and increment pointers
        JP   PE, NEXT      ; return for next byte if BC <> 0
END     RET                ; exit
```

The pointers and counter are set up as before and the code of the marker is placed in A. The current byte is then compared with the marker to see if the end has been reached, in which case the transfer terminates. If there is no match then LDI moves the byte, increments HL and DE, and decrements BC as before but does not automatically repeat. If BC has not reached zero the PARITY/OVERFLOW flag will be set and the routine will jump back for the next byte. If not then it is cleared and the routine ends. The counter and the test for zero are optional. If there is no maximum length to be tested then BC need not be set (it will still be decremented) and line 8 could be JR NEXT.

There are two ways in which these routines can be modified to make them more general. Firstly, the pointers and counter may not be known ahead of time but be calculated by the main program and held as variables. This makes the IMMEDIATE addressing mode impossible and the ABSOLUTE or EXTENDED mode must be used. Secondly, the routine might have to work out the direction of transfer in order to use the appropriate instruction, as mentioned before. The most obvious way to do that is to subtract TO from FROM and test for the sign of the result. Fortunately the HL register pair may be used as a 16 bit accumulater and there is a 16 bit SuBtract with Carry instruction available. The general routine might end up looking like this:-

```
    FROM  DEFW nn          ; pointers and counter
    TO    DEFW nn          ; variables defined in
    COUNT DEFW nn          ;  main program
;
TRNSF LD   HL, (FROM)      ; load pointers
      LD   DE, (TO)        ;
      LD   BC, (COUNT)     ; & counter
      SCF                  ; set carry flag
      CCF                  ; complement (ie. clear) carry flag
      SBC  HL, DE          ; FROM - TO
      LD   HL, (FROM)      ; reload FROM
      JP   M, MVUP         ; jump if result negative
      LDIR                 ; else transfer down
      JR   END             ; & exit
MVUP  LDDR                 ; transfer up
END   RET                  ; exit
```

As you can see the pointers are now passed to the routine from outside and both forms of the transfer instruction are used. SBC is a subtract with carry so the carry flag must be cleared first. FROM has been corrupted by the subtraction and must be reloaded but, as load instructions do not affect the flags, the SIGN flag is still valid.

A similar routine could be written using the LDI and LDD instructions in a more general way with tests for markers or address locations as needed. There are many possible modifications but the essential technique is defined by the nature of the block transfer instructions themselves. Obviously the same effect could be achieved by a combination of load, increment and decrement instructions but the result would be more lines of code and slower results.

NB. :- The transfer process is in fact a copy and the original data is unchanged until it is subsequently overwritten.

The Find Utility program was originally written by Chris Stamboulidis and published in Personal Computer Games (P. 62) in April, 1985. At the time, the utility was located in an unused section of the Communications area of RAM, starting at 7A28H (31272 Decimal).

My contribution to the utility has involved firstly, the addition of cursor positioning before printing, to prevent longer numbers wrapping around the screen, which made reading them a little difficult. Secondly as well as printing out line numbers I added the facility to display the location of the line in memory. Finally, as the program would no longer fit the unused RAM area, I added a relocation routine, to shift the utility to the top of available memory.

The utility enables a search to be made of a BASIC program for a specified string of ASCII characters. Because a detokenisation of the string is carried out, even BASIC keywords such as REM, which are normally stored as single byte tokens, will be located, as will the same string of ASCII characters should they occur between inverted commas following a PRINT statement eg. PRINT" TREMBLE". Any leading spaces in the string specified will be ignored by the routine in its search for a match.

To utilise the command, type PRINT&", followed by the ASCII string of characters to be searched for and then close the inverted commas, as is normally the case eg. PRINT&"REM". Finally press the RETURN key. If occurences of the string are found within the program, the relevant line number and memory location will be printed on the screen. The memory location given, is the first byte of the first character in the line of BASIC and not the location of the first byte of the string being searched for.

I am certain that there are many modifications and enhancements, that can be made to this program, so go to it.

```
###############################################
# THIS SECTION ASSIGNS VALUES TO SOME OF THE    #
# LABELS USED WITHIN THE PROGRAM.               #
###############################################

BUFR EQU  7A9DH        ; BUFFER FOR SEARCH STRING
LEN  EQU  7AD6H        ; CONTAINS LENGTH OF SEARCH STRING
NUM  EQU  79ADH        ; CONTAINS CURRENT LINE NUMBER
MEM  EQU  7A28H        ; CONTAINS MEMORY LOACTION OF LINE
NEXT EQU  79B0H        ; POINTS TO START OF NEXT LINE

###############################################
# THIS SECTION OF THE PROGRAM RELOCATES THE FIND #
# ROUTINE TO THE TOP OF MEMORY                  #
###############################################

     CALL 1B49H        ; DO A NEW
     LD   HL,7AE9H     ; SET START OF BASIC
     LD   (78A4H),HL   ; PROGRAM POINTER
TOPM LD   HL,(78B1H)   ; GET TOP OF MEMORY POINTER
     LD   BC,END-FIND  ; GET LENGTH OF PROGRAM
     PUSH BC           ; SAVE IT
     XOR  A            ; RESET ALL FLAGS
     SBC  HL,BC        ; SUBTRACT PROGRAM LENGTH FROM TOP MEMORY
     LD   (78B1H),HL   ; LOAD NEW TOP MEMORY
     INC  HL           ; INCREASE BY ONE
     PUSH HL           ; SAVE IT
     LD   BC,51        ; RESERVE 50 BYTES OF STRING SPACE
     XOR  A            ; RESET ALL FLAGS
     SBC  HL,BC        ; SUBTRACT STRING SPACE FROM TOP OF MEMORY
```

```
        LD   (78A0H)      ; LOAD START OF STRING SPACE ADDRESS
        DEC  HL           ; DECREASE BY ONE
        LD   (78E8H),HL   ; LOAD ADDRESS OF STACK
        POP  DE           ; RETRIEVE TOP OF MEMORY ADDRESS PLUS ONE
        POP  BC           ; RETRIEVE PROGRAM LENGTH
INIT    LD   A,0C3H       ; LOAD THE JUMP OPCODE
        LD   (7994H)      ; INTO PRINT & VECTOR
        LD   (7995H),DE   ; LOAD VECTOR WITH DESTINATION ADDRESS
        LD   HL,FIND      ; LOAD START ADDRESS OF PROGRAM TO MOVE
        LDIR              ; MOVE IT
        LD   BC,1A18H     ; RETURN
        JP   19AEH        ; TO BASIC

##################################################
# THIS SECTION OF THE PROGRAM IS THE FIND ROUTINE#
##################################################

FIND    INC HL            ; HL POINTS TO SEARCH STRING
        CALL 358CH        ; MOVE STRING TO THE BUFFER
        LD A,(LEN)        ; GET LENGTH OF STRING
        DEC A             ; SUBTRACT ONE
        LD   (LEN),A      ; AND REPLACE IT
        OR   A            ; IF NULL STRING
        JR   Z,EXIT       ; THEN EXIT
        LD   IX,(78A4H)   ; GET START OF PROGRAM
TEST    LD   A,(IX+00H)   ; GET LSB OF POINTER
        OR   A            ; CHECK FOR ZERO
        JR   NZ,CONT      ; IF NOT, THEN CONTINUE
        LD   A,(IX+01H)   ; GET MSB OF POINTER
        OR   A            ; CHECK IF ZERO TOO
        JR   Z,EXIT       ; MUST BE END OF PROGRAM, SO EXIT
CONT    LD   L,(IX+00H)   ; GET POINTER
        LD   H,(IX+01H)   ; TO THE NEXT LINE
        LD   (NEXT),HL    ; SAVE POINTER
        LD   HL,(IX+02H)  ; GET LINE
        LD   HL,(IX+03H)  ; NUMBER
        LD   (NUM),HL     ; SAVE CURRENT LINE NUMBER
        PUSH IX           ; MOVE POINTER
        POP  HL           ; INTO HL
        INC  HL           ; INCREMENT
        INC  HL           ; TO FIRST
        INC  HL           ; BYTE OF
        INC  HL           ; PROGRAM LINE
        LD   (MEM),HL     ; SAVE ADDRESS OF FIRST BYTE IN LINE
        CALL 2B7EH        ; DETOKENISE CURRENT LINE
        LD   DE,79E8H     ; LOCATION OF EXPANDED LINE
PRE     LD   A,(LEN)      ; GET LENGTH OF STRING
        LD   B,A          ; INTO B
        LD   HL,BUFR-1    ; BYTE BEFORE BUFFER
SCAN    INC  HL           ; NEXT BYTE IN STRING
        LD   A,(HL)       ; CHECK IF END
        OR   A            ; OF STRING
        JR   Z,EXIT       ; IF SO, THEN EXIT
        LD   A,(DE)       ; GET BYTE FROM LINE
        OR   A            ; CHECK FOR END OF LINE
        JR   Z,MORE       ; IF SO, THEN NEXT LINE
        INC  DE           ; GET NEXT BYTE
        CP   (HL)         ; CHECK IF SAME AS STRING
        JR   NZ,PRE       ; IF NOT, THEN NEXT BYTE
        DJNZ SCAN         ; CONTINUE UNTIL ALL FOUND
RSME    LD   A,20H        ; ALL FOUND
        CALL 033AH        ; SO PRINT A SPACE
```

```
         LD   HL,(NUM)     ; GET LINE NUMBER
         CALL 0FAFH        ; AND PRINT IT
         LD   A,20H        ; PRINT
         CALL 033AH        ; A SPACE
         LD   HL,(MEM)     ; GET MEMORY LOCATION OF LINE
         CALL 0FAFH        ; AND PRINT IT
         LD   A,(7AAEH)    ; CHECK CURSOR POSITION
         ADD  A,0BH        ; IF NOT ENOUGH
         AND  20H          ; ROOM ON LINE
         JR   NZ,CRTN      ; DO A CARRIAGE RETURN
    MORE LD   IX,(NEXT)    ; GET NEXT LINE
         JR   TEST         ; GO CHECK NEXT LINE
    CRTN LD   A,0DH        ; LOAD CARRIAGE RETURN VALUE
         CALL 033AH        ; PRINT IT
         JR   MORE         ; CONTINUE SEARCH
    EXIT JP   1A19H        ; RETURN TO BASIC
    END  DEFB 0            ; END OF PROGRAM
```

NOTE :- As Larry mentions there are enhancements or mods that could be made and one that springs to mind is for FIND to work properly with LPRINT command. If LPRINT&"string" is used then the printout is without any spaces whatsoever between line numbers and memory addresses. Ed.

DO IT YOURSELF PROJECT

REPLACING VZ 200/300 MEMBRANE KEYBOARD :-

If some of your keys on the VZ do not work or you have to hit them hard (OUCH) to register then you need a replacement MEMBRANE KEYBOARD which is available from Dick Smith stores. They have to be ordered in from spares and it may take a few weeks or longer before it arrives. Below is the part No's. for VZ200/300 M. K'boards.

PART NUMBER - X 73000H0041 - VZ300 MEMBRANE KEYBOARD - Price $22.50.
PART NUMBER - X 72000H0020 - VZ200 MEMBRANE KEYBOARD - Price $22.50.

NOTE :- Check with Dick Smith for correct prices.

Replacing the Memb. K'board is fairly simple. Turn the VZ upside down and remove the six screws holding the two halves of the case together and turn VZ right way up again. Next lift the top half of case up and turn it upside down going towards yourself.

You will see a ribbon cable going from the computer P.C.B (Printed Circuit Board) to the bottom of the keyboard P.C.B. Remove the 12 small screws holding the two halves of the keyboard together being carefull not to lose the small screws in the process.

Next lift the P.C.B. and put it to one side. The Memb. K'board should be visible now which is removed. Before proceeding further get two books about 10mm thick and put a book under each end of the upside down keyboard so the keys are hanging down and not touching the table.

Before putting the new Memb. K'board in which is slightly longer with extra keypads, a small modification has to be carried out to it. One of the extra keypads has to be cut out which covers the hole where the LED goes in. Once done just reassemble being careful not to overtighten the screws. You'll find your keyboard now has a new lease of the life.

NOTE - I've replaced the M. K'board on VZ300 only. The procedure should be the same for VZ200, but don't forget it's M. K'board has a different part No.

****

EDITOR ASSEMBLER TAPE TO DISK CONVERSION UTILITY

- CONVERT YOUR EDITOR ASSEMBLER TO FULL DISK OPERATION -

VZ USER has a conversion package to convert the Dick Smith Editor Assembler (Version 1.2). All SAVES/LOADS etc. to Disk. (Version 1.1 converter coming soon).

Price $15.00 inc. postage and is available from :-
Mark Harwood (Editor) 'VZ USER'
P.O. BOX 154 DURAL NSW AUSTRALIA Phone (02) 651 1413 AH

****

* * FOR SALE * * * * FOR SALE * *

E&F W.P PATCH3.1 - QUICKWRITE W.P

PATCH3.1 - COPYRIGHT - H.V.VZ.U.G.

This single Patch will convert your E & F TAPE WORD PROCESSOR for full DISK use while retaining all TAPE functions. It can be used with 1 or 2 DRIVES. Below are the two Menus.

E)DIT TEXT	L)OAD
C)LEAR TEXT	S)AVE
P)RINT TEXT	D)IR
L)OAD FILE	E)RA
S)AVE FILE	R)EN
V)ERIFY FILE	I)NIT
Q)UIT PROGRAM	1-2) DRIVE 1
D)ISK	M)ENU

Fast SAVING and LOADING of TEXT DATA to and from Disk is provided using Block SAVE or LOAD.

Full instructions are supplied together with a Tape to Disk transfer utility for your E & F Tape Word Processor.

This Patch will work with V1.0 or V1.2 Disk Controller. A STATUS facility has been added for V1.0 DOS owners.

SYSTEM REQUIREMENTS :-
DISK DRIVE + V1.0 OR V1.2 DOS
VZ300 + 16K RAM PACK OR
VZ200 + 18K (16K RAM PACK + 2K)

The price - $10.00, NZ AU$12.00 and is available from :-

HUNTER VALLEY VZ USERS' GROUP
P.O.BOX 161 JESMOND 2299
N.S.W. AUSTRALIA Phone (049)51 2756

* * * NEW NEW NEW * * *

QUICKWRITE WORDPROCESSOR

DISC BASED WORDPROCESSOR
A$40.00

QUICKWRITE WORDPROCESSOR IS SUITABLE FOR THE EXPANDED VZ200 AND VZ300 COMPUTERS.

QUICKWRITE is software on disc, so RAM and ROM PACKS do not have to be plugged and unplugged into the VZ which can cause loose port socket connections.

QUICKWRITE runs on either the LASER or VZ DOS disc controller.

QUICKWRITE saves and loads document text (data) to disc.

FEATURES.

* Fast disc saving and loading of document text (data).
* Automatic periodic saving of data while in typing mode if required.
* Tape saving and loading of data as a backup medium.
* Loading of E&F tape files (data) possible.
* Printer font changes within the data.
* Capitals/lower case software lock on/off.
* Accommodates wide printers - up to 255 columns.
* A Printer/Plotter can also be used.
* Four print justify/wragged modes.
* Adequate operator warnings.
* Labelling of discs allowable, such as date, code etc.
* The usual editing facilities:- Delete, Insert, Find and Replace, Paste, Cut etc.
* Number 1 or number 2 disc drive selection allowed.

* The price of A$40.00. includes surface postage within Australia.

Sold ONLY by VSOFTWAREZ
39 Agnes st., TOOWONG Q/LAND. 4066.
AUSTRALIA.
(07)371 3707.